Journal of Statistical Software

August 2016, Volume 71, Issue 12. *doi: 10.18637/jss.v071.i12*

PerMallows: An R Package for Mallows and Generalized Mallows Models

Ekhine Irurozki
University of the Basque Country

Borja Calvo
University of the Basque Country

Jose A. Lozano
University of the Basque Country

Abstract

In this paper we present the R package **PerMallows**, which is a complete toolbox to work with permutations, distances and some of the most popular probability models for permutations: Mallows and the Generalized Mallows models. The Mallows model is an exponential location model, considered as analogous to the Gaussian distribution. It is based on the definition of a distance between permutations. The Generalized Mallows model is its best-known extension. The package includes functions for making inference, sampling and learning such distributions. The distances considered in **PerMallows** are Kendall's τ, Cayley, Hamming and Ulam.

Keywords: ranking, permutation, Kendall's τ, Cayley, Hamming, Ulam, Mallows, generalized Mallows, learning, sampling, R.

1. Introduction

Permutations are ordered sets of items that arise naturally in many domains, such as genomics (Bader 2011), cryptography, scheduling or computer vision (Ziegler, Christiansen, Kriegman, and Belongie 2012), with that of ranking being the most studied (Burges *et al.* 2005; Mallows 1957).

Probability models for permutation spaces – or permutation models – appear in the same domains as permutations themselves. The first examples come from the area of social choice in which they are still a lively topic (Caragiannis, Procaccia, and Shah 2013). Some of the hottest topics in the research of permutation models are preference learning (Furnkranz and Hullermeier 2013) and learning to rank (Cohen, Schapire, and Singer 1998), which have earned their own space as subfields of machine learning since their commercial applications have increased exponentially in the last years.

There is a great variety of models for permutations, being some of the most popular the Mallows and generalized Mallows models (Mallows 1957), pairwise preference models (Babington Smith 1950; Lu and Boutilier 2011a), the Plackett-Luce's model (PL, Luce 1959) and the Thurstone models (Thurstone 1927).

The **PerMallows** package for the R system (R Core Team 2016) is available from the Comprehensive R Archive Network at `https://CRAN.R-project.org/package=PerMallows`. It is devoted to the Mallows model (MM, Mallows 1957) and the generalized Mallows model (GMM, Fligner and Verducci 1986). The MM is an exponential location model, usually referred to as the Gaussian distribution for permutations. The mode of the distribution is given by one parameter – the central permutation, σ_0 – and the probability of any other permutation increases as we move "closer" to the central permutation. The dispersion parameter, θ, controls how fast this increase happens. The closeness between the central and any other permutation can be measured by any distance for permutations. **PerMallows** includes four distances: Kendall's τ, Cayley, Hamming and Ulam. Their popularity is due to the computational advantages they imply and because they are natural measures of disorder in several domains.

Similarly, in the GMM, the mode is also given as a parameter. The difference is that, instead of having one single dispersion parameter, there are k parameters, each of which affects a particular position of the permutation. By tuning those parameters, we can give more "importance" to discrepancies in certain positions of the permutation.

The literature on the MM includes many general theoretical discussion references (Fligner and Verducci 1988; Critchlow, Fligner, and Verducci 1991; Fligner and Verducci 1986), as well as specific questions such as the analysis of the asymptotic properties of, for example, the length of the longest increasing subsequence of a permutation from a MM (Mueller and Starr 2013; Féray 2013). Moreover, its practical relevance is supported by the application papers in different disciplines such as preference elicitation (Lu and Boutilier 2011b), recommender systems (Sun, Lebanon, and Kidwell 2012), information retrieval (Farah and Vanderpooten 2007), classification (Cheng and Hüllermeier 2009b), label ranking (Cheng and Hüllermeier 2009a) or evolutionary algorithms (Ceberio, Mendiburu, and Lozano 2011). The MM has a great polymorphism and the links with other models have been exploited. The MM and GMM can be easily extended and adapted, as done for nonparametric models (Mao and Lebanon 2008), infinite permutations (Gnedin and Olshanski 2012; Meila and Le Bao 2008), mixture models (D'Elia and Piccolo 2005; Meila and Chen 2010; Murphy and Martin 2003; Lee and Yu 2012) and signed permutations (Arora and Meila 2013).

One of the most special characteristics of the MM is its polymorphism: It can be posed as an exponential function of distances for permutations, as a model for pairwise comparisons and also as a multistage model. It is remarkable the number of ways in which a MM can be learned.

An appealing setting consists of learning a mixture of MM from a set of pairwise observations. While a MM models a homogeneous population (in the sense that the individuals of a population agree in a true consensus), a heterogeneous one (a population in which there are clusters of homogeneous populations) can be modelled with a mixture. An example can be found in Lu and Boutilier (2011a). The authors use a sampling algorithm which is based on the same idea of what we call the multistage sampler in this paper.

The efficient learning and sampling algorithms in **PerMallows** can be the key to developing more complex models, such as mixtures, which combine several MMs.

This is not the first package in the literature to deal with distributions over permutations. **BradleyTerry2** (Turner and Firth 2012), **psychotree** (Strobl, Wickelmaier, and Zeileis 2011) and **prefmod** packages (Hatzinger and Dittrich 2012) model and fit preference data in the form of paired comparisons. In the particular case of the distance-based models, there exist two packages. The **RMallow** (Gregory 2012) package uses an EM algorithm to fit mixtures of MM under the Kendall's τ distance to full or partial rankings, with and without ties as described in Murphy and Martin (2003). On the other hand, the **pmr** package for probability models for ranking data (Lee and Yu 2015) considers Luce's model (Critchlow *et al.* 1991), MM and GMM under the Kendall's τ, Spearman's ρ, Spearman's ρ^2 and foot rule distances. The package is aimed at helping in the analysis of preference data in tasks such as visualizing data and computing descriptive statistics. It implements functions for the maximum likelihood estimation for the parameters of the MM and extensions thereof similar to the GMM.

None of the aforementioned packages offer a wide range of functionalities so as to be considered a complete toolbox for reasoning on permutation data. There is no way of generating permutations from a given model or calculating the probability of a permutation under a certain model, for example. Moreover, none of the packages consider the algebraic machinery necessary for the efficient management of functions over permutations spaces, from basic operations (compositions or factorization into disjoint cycles) to complex combinatorial functions (such as counting or generating permutations).

Regarding the distance metrics, the packages in the literature that deal with distance-based models only consider Spearman's ρ, Spearman's ρ^2, Spearman's foot rule and Kendall's τ, which are the most natural for the preference domain. However, the application of permutations is beyond the preference domain. Cayley is related with the number of swaps and also with the cyclic structure of permutations, Hamming is related with the number of fixed points and Ulam with the longest common subsequence between two permutations. Therefore, Cayley, Hamming and Ulam are more natural in fields such as computer vision, biology, cryptography, matching or card shuffling.

PerMallows, aims to be a compilation of functions for working on distance-based probability models MM and GMM under the Kendall's τ, Cayley, Hamming and Ulam distances. The utilities include the following functions:

- Functions for dealing with MM and GMM: Learning the parameters from a collection of permutations, sampling permutations from a given distribution, computation of the probability of a permutation, of the expectation and marginal distribution. Moreover, several algorithms for each of the tasks are offered, including approximate and exact algorithms.
- Distance related functions: Compute distances between permutations, randomly generate permutations at a given distance, count the number of permutations at a given distance, etc.
- Operations with permutations: Generation of all permutations of a given number of items, inversion, composition, different operators such as, for example, swapping or transposing items, descriptive statistics, factorization of permutations, etc.

Permutations are highly structured combinatorial objects and permutation models cannot obviate this fact. **PerMallows** not only tries to take advantage of statistical tools, but also tries to exploit the algebraic nature of permutations in order to provide efficient sampling and learning algorithms. Moreover, MM and GMM strongly depend on the distances for permutations for several reasons. First, there exists no general algorithm to sample or estimate the parameters of a MM or GMM, since each operation depends on the distance considered. Second, the parameter interpretation also differs regarding the distance considered by the model.

Before presenting the use of the package, different aspects included in the paper need to be discussed. A brief discussion is included in Section 2. For a more detailed discussion we refer the interested reader to Irurozki (2014). Section 2.2 details the probability models and examples of real applications of the models. The particularities of the models under each distance are also enumerated, such as the parameter interpretation and the practical use limits of **PerMallows**. Sections 2.3 and 2.4 introduce several algorithms for learning and sampling respectively. A quick-start guide of **PerMallows** can be found in Section 3. Section 4 concludes the paper.

2. Background

In this section we give the algebraic background for the understanding of the functions in the **PerMallows** package. It is divided into four parts. First, notions on permutations and distance are given. Then, we briefly describe MM and GMM. The last two parts are devoted to the learning and sampling algorithms for the MM and GMM included in **PerMallows**.

2.1. Permutations and metrics

Permutations are bijections of the set of integers $\{1, \dots, n\}$ onto itself. We will denote permutations with Greek letters, mostly π and σ. In the permutation $\sigma = \{2, 4, 1, 3\}$ we will say that item 2 is at position 1 and denote it $\sigma(1) = 2$. The permutation that places every item i at position i is called the identity permutation and it is denoted as $e = \{1, 2, 3, \dots, n\}$. For an excellent reference on combinatorics of permutations see (Bóna 2004).

For every permutation σ, its inverse is denoted as σ^{-1} and defined as $\sigma^{-1}(i) = j \Leftrightarrow \sigma(j) = i$. Two permutations can be composed resulting in a new permutation. The composition operation, denoted $\sigma \circ \pi$ or $\sigma\pi$, is defined as $\sigma\pi(i) = \sigma(\pi(i))$. It is worth noticing that $\sigma\sigma^{-1} = e$.

There are many distance metrics for permutations. Four are considered in **PerMallows**, Kendall's τ, Cayley, Ulam and Hamming. All these metrics are right invariant, meaning that for every permutation τ, σ, π, $d(\sigma, \pi) = d(\sigma\tau, \pi\tau)$. In particular, taking $\tau = \pi^{-1}$ and since $\pi\pi^{-1} = e$, one can, w.l.o.g., write $d(\sigma, \pi) = d(\sigma\pi^{-1}, e)$. When the reference permutation is the identity, we will denote the distance as a one parameter section, $d(\sigma, e) = d(\sigma)$. These two notations will be used interchangeably throughout the paper.

Kendall's τ. This is the most popular choice in the ranking domain. The Kendall's τ distance $d_k(\sigma, \pi)$ counts the number of pairwise disagreements between σ and π, i.e., the number of item pairs that have a relative order in one permutation and a different order in the other. We can equivalently define $d_k(\sigma, \pi)$ as the number of adjacent swaps to convert σ^{-1} into π^{-1}. The maximum value of the Kendall's τ distance between two permutations is $n(n-1)/2$.

The Kendall's τ distance is sometimes called bubble sort distance because $d_k(\sigma)$ equals the number of adjacent swaps that the bubble sort algorithm performs to order the items in σ increasingly. This definition induces the distance decomposition vector $\mathbf{V}(\sigma) = (V_1(\sigma), \ldots, V_{n-1}(\sigma))$, such that $d_k(\sigma) = \sum_{j=1}^{n-1} V_j(\sigma)$ and where $V_j(\sigma)$ equals the number of times that the bubble sort algorithm swaps item $\sigma(j)$. It follows that $0 \leq V_j(\sigma) \leq n - j$ for $1 \leq j \leq n$. Note that $V_j(\sigma)$ is also equal to the number of items smaller than $\sigma(j)$ in the tail of the permutation, and that it can be expressed as follows:

$$V_j(\sigma) = \sum_{i=j+1}^{n} I(\sigma(i) < \sigma(j)), \tag{1}$$

where $I(\cdot)$ denotes the indicator function.

It is worth noticing that there is a bijection between each $\sigma \in S_n$ and each possible $\mathbf{V}(\sigma)$ vector. Therefore, when dealing with the Kendall's τ distance we can use the $\mathbf{V}(\sigma)$ vector as an alternative representation of σ.

As an example, if $\sigma = \{2, 1, 3, 6, 4, 5\}$, then $V(\sigma) = (10020)$ and $d_k(\sigma) = 3$. The conversion from $\mathbf{V}(\sigma)$ to σ and vice versa is supported in **PerMallows** and done in time $O(n^2)$.

Cayley. The Cayley distance $d_c(\sigma, \pi)$ counts the minimum number of swaps (not necessarily adjacent) that have to be made to transform σ into π. The maximum value of the Cayley distance between two permutations is $n - 1$. When the reference permutation is the identity, $d_c(\sigma)$ equals n minus the number of cycles in σ. A cycle in σ is an ordered set $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ such that $\sigma(i_1) = i_2$, $\sigma(i_2) = i_3$, $\ldots$, $\sigma(i_k) = i_1$. It follows that, for every $1 \leq i, j \leq n$ such that $\sigma(i) = j$, then i and j are in the same cycle. The permutation $\sigma = \{1, 5, 6, 4, 2, 3\}$ is written in cycle notation as $(1)(2, 5)(3, 6)(4)$.

Cayley distance $d_c(\sigma)$ can be decomposed into a vector $\mathbf{X}(\sigma)$ of $n - 1$ binary terms, $\mathbf{X}(\sigma) = (X_1(\sigma), \ldots, X_{n-1}(\sigma))$ where:

$$X_j(\sigma) = \begin{cases} 0 & \text{iff } j \text{ is the largest item in its cycle in } \sigma, \\ 1 & \text{otherwise.} \end{cases} \tag{2}$$

It follows from the definition that $d_c(\sigma) = \sum_{j=1}^{n-1} X_j(\sigma)$. Given $\sigma = \{2, 1, 3, 6, 4, 5\}$, then $X(\sigma) = (10011)$ and $d_c(\sigma) = 3$. Contrary to the Kendall's τ distance, there is no bijection between every possible $\mathbf{X}(\sigma)$ vector and $\sigma \in S_n$: although each $\sigma \in S_n$ has one unique decomposition vector $\mathbf{X}(\sigma)$, the opposite is not necessarily true.

Generating uniformly at random a permutation σ consistent with a given $\mathbf{X}(\sigma)$ can be done by slightly adapting the well known Fisher-Yates-Knuth shuffler (Irurozki 2014). This algorithm, as well as the conversions from σ to $\mathbf{X}(\sigma)$, is also supported in **PerMallows**, all of which have time complexity $O(n)$.

Hamming. The Hamming distance $d_h(\sigma, \pi)$ counts the number of positions that disagree in the two permutations. The maximum value of the Hamming distance between two permutations is, therefore, n. It is worth noticing that there is no pair of permutations σ and π such that $d(\sigma, \pi) = 1$.

The Hamming distance is closely related to the concepts of fixed and unfixed points. A fixed point in σ is a position i where $\sigma(i) = i$, while the opposite is an unfixed point. The Hamming distance to the identity, $d_h(\sigma)$, counts the number of unfixed points in σ. This leads to the decomposition vector of the Hamming distance, $\mathbf{H}(\sigma) = (H_1(\sigma), \ldots, H_n(\sigma))$:

$$H_j(\sigma) = \begin{cases} 0 & \text{iff } \sigma(j) = j, \\ 1 & \text{otherwise.} \end{cases} \quad (3)$$

Consequently, $d_h(\sigma) = \sum_{j=1}^{n} H_j(\sigma)$ (note that every $\sigma \in S_n$ has a unique $\mathbf{H}(\sigma)$ but the opposite is not necessarily true). Given $\sigma = \{2,1,3,6,4,5\}$, then $\mathbf{H}(\sigma) = (110111)$ and $d_h(\sigma) = 5$.

The conversion from σ to $\mathbf{H}(\sigma)$ and the generation uniformly at random of σ consistent with a given $\mathbf{H}(\sigma)$ are both supported in **PerMallows** and have complexity $O(n)$.

Ulam. The Ulam distance $d_u(\sigma, \pi)$ counts the length of the complement of the longest common subsequence (LCS) in σ and π, i.e., the number of items which are not part of the LCS. The maximum value of the Ulam distance between two permutations is $n - 1$. If the reference permutation is the identity, $d_u(\sigma)$ equals n minus the length of the longest increasing subsequence (LIS).

The classical example to illustrate the Ulam distance $d_u(\sigma, \pi)$ considers a shelf of books in the order specified by σ (Diaconis 1988). The objective is to order the books as specified by π with the minimum possible number of movements, where a movement consists of taking a book and inserting it in another position (delete-insert). The minimum number of movements is exactly $d_u(\sigma, \pi)$.

For example, given $\sigma = \{2,1,3,6,4,5,7\}$, the length of the LIS is 4 and, therefore, $d_u(\sigma) = 3$. Note that there is not just a single LIS, items 2367 form a LIS, and so do items 1367.

The computation of the Ulam distance between two given permutations is included in **PerMallows**. It has complexity $O(n \, log \, l)$ where l is the length of the longest increasing subsequence.

Counting and generating permutations

The random generation of permutations is a problem of interest in many disciplines. The uniformly at random generation, for example, can be efficiently carried out with the well known Fisher-Yates shuffle (also known as Knuth shuffle). More restrictive version of the problem are the following related questions, which are addressed in the following lines:

- Given the number of items n and a distance d, how many permutations are there at distance d from the identity, e?
- Given the number of items n and a distance d, generate uniformly at random a permutation at the given distance from e.

Due to their right invariant property, the number of permutations at distance d from the identity, denoted $S(n, d)$, is the same as the number of permutations at distance d from any $\sigma \neq e$. There is no closed expression for $S(n, d)$ for any of the metrics in this paper. Fortunately, these sequences appear in the On-Line Encyclopedia of Integer Sequences (OEIS) (Sloane 2009) for every metric considered in **PerMallows** for different values of n and d, as well as some recursions to obtain them.

Kendall's τ. The number of permutations at every possible Kendall's τ distance, $S_k(n,d)$ can be found at the OEIS (Sloane 2009, sequence A008302). Its computational cost is $O(n^3)$. The uniformly at random generation of a permutation at a given Kendall's τ distance, if the values for $S_k(n,d)$ are given, can be carried out in $O(n^2)$ (Irurozki 2014, page 38).

Cayley. The number of permutations at a given Cayley distance are given by Stirling numbers of the first kind (Sloane 2009, sequence A008275) and **PerMallows** can compute it in $O(n^2)$. The uniformly at random generation can be done by adapting the recurrence used in the counting process, as shown in (Irurozki 2014, page 54), running in time $O(n)$.

Hamming. Counting and generating permutations at a given Hamming distance, which is related to the notion of derangements, are computationally efficient operations that **PerMallows** can compute in $O(n)$ (Irurozki 2014, page 17), (Sloane 2009, sequence A000166).

Ulam. Both processes of counting (Sloane 2009, sequence A126065) and generating permutations at a given Ulam distance are related with the Ferrers diagrams (FD) and the Standard Young Tableaux (SYT). The link between SYT and permutations is given by the celebrated Robinson-Schensted-Knuth (RSK) correspondence. A self contained explanation can be found in (Irurozki 2014, page 110). The complexity of the counting and generating processes is equivalent to the complexity of enumerating the partitions of n, which grows sub-exponentially with n (Hardy and Ramanujan 1918).

Counting permutations at a given distance is a crucial operation, not only for the random generation of permutations, but also for the learning and sampling processes. Among the distances considered, Ulam is the most demanding from a computational perspective. **PerMallows** includes functions to preprocessing a problem in order to speed these operations up, as shown in page 25.

2.2. Probability distributions over permutations

This section introduces the Mallows model (Mallows 1957) and its most popular extension, the generalized Mallows model (Fligner and Verducci 1986).

Mallows model

The MM was one of first probability models proposed for rankings or permutations. However, it is still one of the most used models in both theoretical and application papers. It is an exponential model defined by a central permutation σ_0 and a spread (or dispersion) parameter θ. When $\theta > 0$, σ_0 is the mode of the distribution, i.e., the permutation with the highest probability. The probability of any other permutation decays exponentially as its distance to the central permutation increases. The spread parameter controls how fast this fall happens.

The probability of a permutation under this model can be expressed as follows:

$$p(\sigma) = \frac{\exp(-\theta d(\sigma, \sigma_0))}{\psi(\theta)},$$

where $\psi(\theta)$ is the normalization constant. The distance can be measured in many ways, including Kendall's τ, Cayley, Hamming and Ulam distances. The MM under the Kendall's τ distance is also known in the literature as the Mallows ϕ model (Critchlow *et al.* 1991).

Regardless of the distance, the central permutation is the location parameter. When the dispersion parameter θ is greater than 0, then σ_0 is the mode and, as θ increases the distribution gets sharper. On the other hand, with $\theta = 0$ we obtain the uniform distribution and when $\theta < 0$ then σ_0 is the anti-mode, i.e., the permutation with the lowest probability.

The MM is very attractive for researchers and practitioners because of its simple definition and because it is a realistic model in several domains. Its most remarkable drawback is the computational difficulty of working with it since, in general, there are no closed forms for the normalization constant. In addition, there is neither a general approach to sample and learn from the model, nor to express the expectation of the distance. Moreover, these operations strongly depend on the distance for permutations considered by the model.

Generalized Mallows model

This extension of the MM tries to break the restriction of every permutation at the same distance having the same probability value. Instead of one single spread parameter, the GMM requires the definition of k parameters θ_j, each affecting a particular position of the permutation. This allows modelling a distribution with more emphasis on the consensus of certain positions of the permutation while having more uncertainty about some others.

An example of such a situation is as follows. Let σ_0 and π_1 be two rankings that differ only in the first and second ranked items, and let σ_0 and π_2 differ only in the last two ranked items. A MM centred around σ_0 will assign equal probability to π_1 and π_2 since:

$$d(\sigma_0, \pi_1) = d(\sigma_0, \pi_2) \Rightarrow p(\pi_1) = p(\pi_2).$$

However, it is reasonable to think that since σ_0 and π_2 differ in the last positions, their disagreement is not as notorious as the disagreement of σ_0 and π_1. One could expect that $p(\pi_1) < p(\pi_2)$. This situation can be modelled with the GMM by setting the dispersion parameters, for example, as follows:

$$\theta_1 > \theta_2 > \theta_3 > \ldots > \theta_k.$$

Not every distance that can be used in the MM can also be used in GMM, since GMM requires the distance to be decomposed in k terms as follows:

$$d(\sigma, \sigma_0) = d(\sigma\sigma_0^{-1}) = \sum_{j=1}^{k} S_j(\sigma\sigma_0^{-1}). \tag{4}$$

For any distance that decomposes as the above equation, the GMM is defined as follows:

$$p(\sigma) = \frac{\exp(\sum_{j=1}^{k} -\theta_j S_j(\sigma\sigma_0^{-1}))}{\psi(\boldsymbol{\theta})},$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ and $\psi(\boldsymbol{\theta})$ is the normalization constant. **PerMallows** considers the GMM for the Kendall's τ, Cayley and Hamming distances. The GMM under the Kendall's τ distance is known in the literature as Mallows ϕ component model.

It is worth noticing that if the $S_j(\sigma)$ terms are independent for a uniformly at random chosen permutation, then the Mallows distribution is factorizable.

The following lines briefly introduce the particularities of the models under each of the distances considered. We have claimed that the decision on which distance to use depends strongly on the domain. As a general rule, we can state that a MM or GMM based on a particular distance can be used to model data in a domain if that distance is a natural measure of dissimilarity in that domain, proof of which can be found in Caragiannis *et al.* (2013).

Under the MM, the dispersion parameter θ is a measure of the consensus in the population, i.e., the larger θ, the closer the permutations are. Under the GMM, the dispersion parameters θ_j have a different interpretation depending on the distance that is being considered, which will be detailed in each case.

It is worth noticing that the dispersion parameters are not universal measures of spread (Schader 1991). This means that if we consider any of these models under two different distances, the dispersion parameters are not a comparable measure of uniformity since, for different metrics, the dispersion parameters are in different scales.

Kendall's τ. The dispersion parameter $\boldsymbol{\theta}$ in the GMM is a $(n-1)$-dimensional vector. The distribution and the normalization constant can be factorized. They, as well as the expected value of the Kendall's τ distance and V_j under the MM and GMM, can be evaluated efficiently (Fligner and Verducci 1986; Irurozki 2014).

Let σ be a permutation sampled from a GMM under the Kendall's τ distance, with parameters $\boldsymbol{\theta}$ and σ_0, where $\sigma_0(j) = i$. The dispersion parameter θ_j is related to position j in σ in the sense that the larger θ_j, the larger the probability of $\sigma(j) \leq i$. In the ranking domain, when permutations are interpreted as rankings this means that item j is ranked in the first i positions with high probability.

Cayley. The dispersion parameter $\boldsymbol{\theta}$ in the GMM is a $(n-1)$-dimensional vector. The distribution, the normalization constant, the expectation and the probability of each term $X_j(\sigma)$ are factorizable and computationally cheap. Although the factorization was introduced in Fligner and Verducci (1986), it includes some typos, the interested reader can find the corrected version in Irurozki (2014).

Let σ be a permutation from a GMM under the Cayley distance, with parameters $\boldsymbol{\theta}$ and σ_0, where $\sigma_0(j) = i$. The larger θ_j, the larger the probability that $\sigma(i) \leq j$. Regarding the position where item j lies in σ, we can state that the larger θ_j, the larger the probability that $\sigma^{-1}(j) \in \{\sigma_0^{-1}(1), \ldots, \sigma_0^{-1}(j-1)\}$.

Hamming. The dispersion parameter $\boldsymbol{\theta}$ in the GMM is an n-dimensional vector. Contrary to the previous cases, the MM and GMM are not factorizable when the Hamming distance is considered. However, its symmetry yields computationally efficient expressions for the normalization constant, the expectation and the probability of each term $H_j(\sigma)$ (Irurozki 2014).

Let σ be a permutation from a GMM under the Hamming distance, with parameters $\boldsymbol{\theta}$ and σ_0, where $\sigma_0(j) = i$. The larger θ_j, the larger the probability that $\sigma(i) = j$.

Ulam. The Ulam distance has no natural decomposition of the form of Equation 4 and, thus, the GMM cannot be defined under this metric. Moreover, the MM can not be factorized, as far as the authors know. However, an efficient way of computing $\psi(\theta)$ and the expected value of the distance can be found in Irurozki (2014). It relies on the computation of the number of permutations of n items at each possible Ulam distance, $S_u(n, d)$, which means that the complexity of computing $\psi(\theta)$ or $E[d_u]$ is equivalent to that of computing $S_u(n, d)$.

2.3. Model fitting

In this section we deal with the maximum likelihood estimation (MLE) of the parameters of the distribution given a sample of m i.i.d. permutations $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$, i.e., the problem of learning a model from a data sample. The maximum likelihood estimators are different for MM and GMM. Moreover, their expression differs regarding the distance on the permutations considered. In this way, we will describe the maximum likelihood estimation for each model and distance separately. Finally, we will introduce the algorithms provided in **PerMallows**.

Mallows model

For the MM, the maximum likelihood estimation of the parameters can be carried out in two independent stages, which are as follows.

1. The first one deals with the MLE for the central permutation, $\hat{\sigma}_0$. Estimating $\hat{\sigma}_0$ of a sample under a MM which considers the Kendall's τ (respectively Cayley, Hamming and Ulam) distance is equivalent to finding the permutation that minimizes the sum of the Kendall's τ (respectively Cayley, Hamming and Ulam) distance to the permutations in the sample. For the distances considered here, only the case of Hamming has polynomial complexity, as far as the authors are aware.

2. The second one estimates the MLE for the dispersion parameters for the given $\hat{\sigma}_0$. The expression for $\hat{\theta}$ given $\hat{\sigma}_0$ has no closed expression for any of the distances considered. However, since the first derivative of the log-likelihood is monotonic, it can be solved efficiently with numerical methods such as Newton-Raphson.

The complete statements of the problems can be found in Mandhani and Meila (2009) and Irurozki (2014).

Generalized Mallows model

For the GMM, the exact maximum likelihood estimation of the parameters can not be carried out in two stages and, thus, all the parameters must be simultaneously estimated. The MLE for the parameters of a GMM under the Kendall's τ, Cayley and Hamming distances can be found in Mandhani and Meila (2009); Irurozki (2014).

Learning algorithms under each distance

In this section we briefly describe the learning algorithms implemented in **PerMallows** for learning MM and GMM. Since the exact estimation of the parameters, as described in the previous section, can be computationally intractable under certain conditions, approximate algorithms are also included for most of the cases.

The crucial speed-up stage in the approximate algorithms for both MM or GMM is the approximation of the central permutation. These algorithms divide the learning process in two stages, first approximate the central permutation $\hat{\sigma}_0$ and second estimate the dispersion parameters for the given $\hat{\sigma}_0$.

Kendall's τ. For the Kendall's τ distance, **PerMallows** includes the approximate estimation of MM and GMM. The estimated $\hat{\sigma}_0$ is the result of the Borda algorithm (Borda 1781). This well-known algorithm is fast ($O(mn)$) as well as a good approximation (factor 5 approximation of the optimal ranking (Coppersmith, Fleischer, and Rurda 2010) and also an asymptotically optimal estimator of the real central permutation (Fligner and Verducci 1986)).

The interested reader can find in Ali and Meila (2012) the description and performance analysis of several exact and approximate algorithms for the MLE for the consensus permutation under the MM. The referenced paper is an extensive work which describes and compares 104 algorithms and combinations thereof, including Borda and Dwork, Kumar, Naor, and Sivakumar (2001). The problem of the exact MLE for the parameters of a GMM is addressed in Mandhani and Meila (2009).

Cayley. The **PerMallows** package includes exact and heuristic algorithms for the MLE for the parameters of both MM and GMM. The exact algorithm gets faster as the consensus of the sample increases. The approximate algorithm uses a heuristic step to create an initial solution which is further improved using a Variable Neighborhood Search (VNS) to iteratively improve it (Irurozki 2014).

Hamming. There exists an exact polynomial time algorithm to fit the maximum likelihood parameters of the MM under the Hamming distance which is included in **PerMallows**, so an approximate algorithm is not needed.

For the GMM under the Hamming distance, **PerMallows** also includes a polynomial time algorithm. Although not an exact algorithm, it has been shown to be a good approximation in theory (it is an asymptotically unbiased estimator of the real solution) and in practice. An exhaustive review of MM and GMM under the Hamming distance can be found in Irurozki (2014).

Ulam. **PerMallows** supports the approximate learning of parameters of the MM under the Ulam distance.

The approximate consensus given a collection of permutations is approached as a set-median problem. The set-median permutation is the permutation in the sample that minimizes the sum of the distances to the rest of the permutations in the sample. Note that this problem can be solved in time $O(m^2 nlog\, n)$ for samples of m permutations.

The expression of the MLE for the dispersion parameter given the consensus permutation and further details are given in Irurozki (2014). The package includes the possibility of precomputing the number of permutations, which can be used to efficiently deal with the Ulam distance for large permutations of more than 80 items, see page 25.

2.4. Sampling

PerMallows implements three different algorithms for generating permutations from a given distribution: The Distances and the multistage algorithms, which are exact, and the Gibbs algorithm (a traditional approximate algorithm).

Distances sampling algorithm

The Distances sampling algorithm can generate samples from the MM under the Kendall's τ, Cayley, Hamming and Ulam metrics. It is based on the facts that, first, under the MM every permutation at the same distance from the central permutation has the same probability, and secondly, the numbers of permutations at each distance, denoted as $S(n, d)$ can be computed (see Section 2.1.1). Since our first claim does not hold for the GMM, it follows that this sampler can be used only to sample from the MM. Consequently, the probability of obtaining a permutation at distance d under the MM is as follows:

$$p(\sigma | d(\sigma, \sigma_0) = d) \propto S(n, d) \exp(-\theta d), \tag{5}$$

The process of simulating from the distribution can be carried out in three stages:

1. Randomly select the distance at which the permutation will lie using Equation 5.

2. Pick, uniformly at random, a permutation π at distance d from the identity permutation e, i.e., $d(\pi) = d$. This step relies on the u.a.r. generation of a permutation at a given distance discussed in Section 2.1.1.

3. In the case that $\sigma_0 = e$, then π is the output. Otherwise, the invariance property lets us obtain $\sigma = \pi\sigma_0$, since $d = d(\pi) = d(\pi\sigma_0, \sigma_0) = d(\sigma, \sigma_0)$.

The complexity of the algorithm depends on the considered distance. See Section 2.1.1 for the complexity of counting and generating permutations.

We can conclude that this is a quick as well as precise algorithm for the simulation of the MM. However, it has some limitations. First, it does not work with the GMM. Also, it is infeasible to keep $S(n, d)$ for values of $n > 150$ (see Table 1). In these situations we can use the multistage sampling algorithm.

Multistage sampling algorithm

The multistage sampling algorithm generates samples from the MM and GMM under the Kendall's τ, Cayley and Hamming metrics. This algorithm divides the sampling process into three stages, namely:

1. Randomly generate a distance decomposition vector, $\mathbf{S}(\pi)$, for the Kendall's τ, Cayley or Hamming distances. Details can be found in (Irurozki 2014, pages 41, 59 and 97 respectively).

2. Generate a permutation π uniformly at random consistent with the given distance decomposition vector $\mathbf{S}(\pi)$.

3. In the case that $\sigma_0 = e$, then π is output. Otherwise, output $\sigma = \pi\sigma_0$.

The complexity of step 1 is $O(n)$ for the Kendall's τ and Cayley distances and $O(n^2)$ for Hamming distance. The complexity of step 2 for each distance is detailed in Section 2.1. This method can generate permutations from both MM and GMM and it can efficiently handle distributions on permutations of large n. See Table 1 for the practical limits.

			Kendall	Cayley	Hamming	Ulam
Counting and generating			150	150	150	100
MM	Sampling	Distances	150	150	150	100
		Multistage	500	500	200	×
		Gibbs	500	500	500	×
	Learning	Exact	×	250	90	×
		Approximate	80	250	×	100
GMM	Sampling	Distances	×	×	×	×
		Multistage	500	500	200	×
		Gibbs	500	500	500	×
	Learning	Exact	×	500	×	×
		Approximate	500	500	250	×

Table 1: Applicability restrictions of the algorithms for each of the supported models.

Gibbs sampling algorithm

The Gibbs sampler is a Markov chain Monte Carlo algorithm based on sampling a Markov chain whose stationary distribution is the distribution of interest. Therefore, we have adapted this algorithm to generate samples from approximate distributions of both MM and GMM.

The Gibbs sampler generates permutations by moving from one solution to another permutation which is close to the first one, where the definition of *close* is related to the distance for permutations considered in the model. The initial samples are discarded (burn-in period) until the Markov chain approaches its stationary distribution, and so samples from the chain are samples from the distribution of interest. Then, the above process is repeated until the algorithm generates a given number of permutations.

The Gibbs algorithm has complexity $O(n)$ for every distance considered with the exception of the MM under the Ulam distance, for which the complexity is $O(n\ log\, n)$. Therefore, it is in general the fastest algorithm. For those users tempted by this theoretical nice time performance, we should emphasize the fact that this an approximate sampling algorithm. Further details can be found in Irurozki (2014)

2.5. Summary

PerMallows provides a complete toolbox for working with permutations, the Mallows and the generalized Mallows models. These probability models need the definition of a metric for permutations and **PerMallows** considers four different distances: Kendall's τ, Cayley, Hamming and Ulam. The package includes three sampling algorithms and two learning algorithms. However, not every algorithm can be applied to the models under every metric. As a summary, we have included in Table 1 the applicability of the algorithms for each metric as well as the maximum number permutation length that each function can handle.

3. The PerMallows package

In this section we show how to use the **PerMallows** package, from basic operations such as measuring distances, to more complex task of learning and sampling. Section 3.2 illustrates how to deal with the MM and GMM. For the sake of reproducibility, set first the random seed.

```
R> set.seed(1)
```

3.1. Permutations

Generation The most basic function consists in creating permutations, which are vectors containing the first n natural numbers and where each item appears once and once only. They can be defined by hand as follows:

```
R> sigma <- c(1, 5, 6, 4, 2, 3)
R> sigma

[1] 1 5 6 4 2 3
```

The validity of a vector as a permutation can be checked with the function `is.permutation`.

```
R> is.permutation(perm = sigma)

[1] TRUE

R> is.permutation(c(0, 1, 5, 4, 2, 3))

[1] FALSE

R> is.permutation(c(1, 8, 9, 2, 5, 3))

[1] FALSE
```

The identity permutation is that which maps every item i to position i. It can be created with the `identity.permutation` function. The number of items in the permutations, n, is denoted `perm.length` in **PerMallows**.

```
R> identity.permutation(perm.length = 6)

[1] 1 2 3 4 5 6
```

The generation of a permutation uniformly at random is supported via the `runif.permutation` function.

```
R> runif.permutation(n = 2, perm.length = 6)

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    2    3    5    6    1    4
[2,]    6    1    5    3    4    2
```

The generation of the set of every possible permutation of n items is carried out with the `permutations.of` function. Recall that the number of permutations of n items increases factorially with n and it is, thus, computationally expensive to generate every permutation of $n \geq 10$. By default, the `alert` argument is set to TRUE. When `alert` is `TRUE` and n is greater than 9, an alert message is shown.

```
R> permutations.of(perm.length = 3, alert = FALSE)

     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    1    3    2
[3,]    2    1    3
[4,]    2    3    1
[5,]    3    1    2
[6,]    3    2    1

R> permutations.of(perm.length = 10)

Do you really want to generate all  3628800  permutations? (Y/N)
[1] "Process cancelled"
```

The collection of permutations generated is stored as a 2-dimensional matrix. It is also possible to read such a matrix from disk using the function `read.perms`, which also checks if every row is a valid permutation.

```
R> path = system.file("test.txt", package = "PerMallows")
R> sample = read.perms(path)
```

Together with the **PerMallows** package, we provide some small datasets that will be used as running examples throughout this reference manual.

Another way of generating permutations is by ranking the ratings of a set of items. Suppose we have the results time elapsed by five runners in three different races. An example of such a file is given in `data.order`. After reading the file, we can get the ranks of each student with the function `order.ratings`.

```
R> data("data.order")
R> data.order

   V1   V2  V3  V4   V5
1 0.1 4.20 2.0 9.4 9.00
2 6.3 2.11 0.1 5.7 4.00
3 9.0 4.50 7.1 6.3 0.21

R> order.ratings(ratings = data.order)

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    2    5    4
[2,]    3    2    5    4    1
[3,]    5    2    4    3    1
```

Operations Two permutations can be composed and the result is a permutation.

```
R> sigma <- c(1, 5, 6, 4, 2, 3)
R> pi <- c(3, 5, 1, 2, 6, 4)
R> compose(perm1 = sigma, perm2 = pi)

[1] 6 2 1 5 3 4
```

Be aware that the composition is not commutative.

```
R> compose(perm1 = sigma, perm2 = pi)

[1] 6 2 1 5 3 4

R> compose(perm1 = pi, perm2 = sigma)

[1] 3 6 4 2 5 1
```

Our implementation of the composition allows one of the arguments (first or second, but not both at the same time) to be a collection of permutations. In this case, every permutation in the sample is composed with the permutation in the other argument, resulting in a new collection of permutations.

```
R> tau <- c(2, 1, 3, 4)
R> data("perm.sample.small")
R> compose(perm1 = perm.sample.small, perm2 = tau)

     V2 V1 V3 V4
[1,]  3  1  2  4
[2,]  1  2  3  4
[3,]  4  1  3  2
[4,]  2  1  4  3
[5,]  3  2  4  1
```

A useful summary of a sample is given by the number of permutations in the sample in which item i appears at position j. This is usually denoted as the frequency matrix or first order marginal matrix. The current package supports it via the `freq.matrix` function. We will illustrate the `freq.matrix` function using the well known APA dataset, a version of which is included in this package. This dataset has been largely used in the literature. In particular, one can find in Diaconis (1989) a spectral analysis that takes into account the first order marginal of the dataset.

The American Psychological Association (APA) dataset includes 15449 ballots of the election for the president in 1980 (Diaconis 1989). Each voter ranked at least one of the five candidates. Along with this package we distribute the 5738 ballots that ranked all the five candidates under the name of `data.apa`. Its marginal matrix is computed as follows:

```
R> data("data.apa")
R> freq.matrix(perm = data.apa)

          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.1835134 0.2647264 0.2288254 0.1746253 0.1483095
[2,] 0.1350645 0.1876961 0.2466016 0.2467759 0.1838620
[3,] 0.2804113 0.1673057 0.1382015 0.1829906 0.2310910
[4,] 0.2042524 0.1693970 0.1897874 0.2028581 0.2337051
[5,] 0.1967585 0.2108749 0.1965842 0.1927501 0.2030324
```

Under the ranking interpretation, $\sigma(i) = j$ denotes that item i is ranked at position j. It follows that the first column counts the proportion of the votes in which each candidate was ranked as the favorite. We can see that the third candidate was chosen as the best alternative by the majority of the voters and thus won the election.

Another basic operation for permutations is inversion. The inverse of a permutation can be obtained using the function `inverse.perm`.

```
R> inverse.perm(perm = sigma)

[1] 1 5 6 4 2 3
```

The argument `perm` can be a single permutation or a collection of permutations. It is worth noting that the inverse of the inverse of any permutation is itself.

```
R> inverse.perm(inverse.perm(c(1, 4, 5, 3, 2)))

[1] 1 4 5 3 2
```

The present package also includes operators for the manipulation of permutations. The `swap` function, for example, swaps two prescribed items.

```
R> swap(perm = identity.permutation(6), i = 1, j = 3)

[1] 3 2 1 4 5 6
```

The `insert` function takes an item from position `i` and inserts it after position `j`.

```
R> insert(perm = identity.permutation(6), i = 5, j = 2)

[1] 1 2 5 3 4 6
```

An inversion at position $1 \leq i < n$ occurs when the items at positions i and $i+1$ are swapped.

```
R> inversion(perm = identity.permutation(6), i = 1)

[1] 2 1 3 4 5 6
```

Distances We will now show how to deal with functions related with the distances between permutations included in this package. These functions include the argument `dist.name` which is one of the following: `kendall`, `cayley`, `hamming` or `ulam`. They can also be denoted by their initial letter: `k`, `c`, `h` and `u`.

The `distance` function computes the distance between `perm1` and `perm2` for a given metric.

```
R> pi

[1] 3 5 1 2 6 4

R> sigma

[1] 1 5 6 4 2 3

R> distance(perm1 = sigma, perm2 = pi, dist.name = "cayley")

[1] 4

R> distance(perm1 = sigma, perm2 = pi, dist.name = "h")

[1] 5
```

The arguments `perm2` and `dist.name` can be omitted. In that case, `perm2` is assumed to be the identity and `dist.name` is `kendall`.

As we have seen, the Kendall's-τ, Cayley and Hamming distances from a permutation to the identity can be decomposed in a vector. Clearly, this decomposition is different regarding the metric in question. In particular, the decomposition of the Kendall's-τ distance is a vector of $n-1$ integer terms, for the Cayley distance it is a binary vector of $n-1$ terms and for the Hamming distance it is a binary vector of n terms. These decompositions are computed by `perm2decomp`:

```
R> sigma

[1] 1 5 6 4 2 3

R> v.vector <- perm2decomp(perm = sigma, dist.name = "k")
R> v.vector

[1] 0 3 3 2 0

R> x.vector <- perm2decomp(perm = sigma, dist.name = "c")
R> x.vector

[1] 0 1 1 0 0
```

```
R> h.vector <- perm2decomp(perm = sigma, dist.name = "h")
R> h.vector

[1] 0 1 1 0 1 1
```

Given that there are possibly many longest increasing subsequences in a permutation, there is no decomposition of the Ulam distance. The possible values for the `dist.name` are therefore, `kendall`, `cayley` and `hamming`, being the default value `kendall`.

The **PerMallows** package can also perform the inverse operation, that is, given a decomposition vector and a metric, obtain a permutation consistent with the vector, `decomp2perm`. Recall that for Cayley and Hamming there are possibly many permutations with a particular decomposition. In these situations, the function recovers uniformly at random one of the permutations consistent with the decomposition vector.

```
R> decomp2perm(vec = v.vector, dist.name = "kendall")

[1] 1 5 6 4 2 3
```

```
R> decomp2perm(vec = x.vector, dist.name = "cayley")

[1] 1 3 4 2 5 6
```

```
R> decomp2perm(vec = h.vector, dist.name = "hamming")

[1] 1 5 6 4 2 3
```

Related with the Cayley distance, the **PerMallows** package implements a function to obtain the list of the cycles in which the permutation decomposes.

```
R> cycles <- perm2cycles(perm = sigma)
```

The `cycle2str` function can be used in order to display the cycles in a user-friendly way.

```
R> cycle2str(cycles)

(1)(5 2)(6 3)(4)
```

Also, the inverse operation, consisting of building a permutation given the list of cycles, is supported.

```
R> cycles2perm(cycles = cycles)

[1] 1 5 6 4 2 3
```

PerMallows also includes a function to count the number of permutations of `perm.length` items at distance `dist.value` for a given distance.

```
R> count.perms(perm.length = 6, dist.value = 2, dist.name = "ulam")
```

```
[1] 181
```

The Cayley distance d is related to the number of cycles c by the following relation: $n = c+d$. Therefore, the number of permutations with `perm.length` items and `num.cycles` cycles can be obtained as follows.

```
R> num.cycles <- 1
R> len <- 6
R> count.perms(perm.length = len, dist.value = len - num.cycles,
+    dist.name = "c")
```

```
[1] 120
```

The notions of fixed points and derangements are related to the Hamming distance. A fixed point is a position such that $\sigma(i) = i$. A derangement is a permutation with no fixed point and its Hamming distance to the identity equals n. In **PerMallows** the number of derangements of a permutation of `perm.length` items can be obtained as follows.

```
R> count.perms(perm.length = len, dist.value = len, dist.name = "hamming")
```

```
[1] 265
```

The generation of random permutations at a prescribed distance is supported by the `rdist` function, which generates `n` permutations of `perm.length` items at distance `dist.value` for a particular metric.

```
R> rdist(n = 4, perm.length = 5, dist.value = 3, dist.name = "ulam")
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    5    4    2    1    3
[2,]    3    1    5    4    2
[3,]    4    5    3    1    2
[4,]    4    3    2    1    5
```

This function can be used to generate permutations with a given number of cycles.

```
R> cycles <- 2
R> rdist(n = 3, perm.length = len, dist.value = len - cycles, "cayley")
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    6    1    5    2    3    4
[2,]    2    1    4    6    3    5
[3,]    4    2    1    6    3    5
```

Similarly, it can be used to generate derangements.

```
R> rdist(n = 3, perm.length = len, dist.value = len, "hamming")

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    3    6    1    5    4    2
[2,]    3    4    1    6    2    5
[3,]    5    1    4    2    6    3
```

3.2. Distributions on permutations

In this section we show how to deal with the Mallows and Generalized Mallows models. We show functions for making inference, learning and sampling both models.

Bear in mind that MM can be used with every metric for permutation considered in this manuscript (Kendall's-τ, Cayley, Hamming and Ulam), while the GMM can only be used with Kendall's-τ, Cayley and Hamming. Remember also that, for distributions on permutations of n items, the dispersion parameter vector $\boldsymbol{\theta}$ has n-1 terms when the distance is either Kendall's-τ or Cayley, and n when the distance is Hamming.

Parameter fitting The estimation of the parameters of the MM and GMM is carried out with separate functions, `lmm` and `lgmm` respectively. The following lines illustrate how to fit the parameters of the APA dataset which has been used in previous examples. The most natural distance for ranking data is the Kendall's-τ and, consequently, the distance-based probability models for voting data are based on the Kendall's-τ. The MM, which by default considers the Kendall's-τ distance, can be used to model the APA dataset as follows:

```
R> lmm(data.apa)

$mode
[1] 1 5 2 4 3

$theta
[1] 0.07218874
```

The mode of the distribution σ_0 is the average ranking, i.e., the ranking that minimizes the sum of the distances to the rankings in the sample. The dispersion parameter is a degree of the consensus of the population. Recall that the dispersion parameters are only comparable for any two models that consider the same distance, as explained in Section 2.2.

The Hamming distance, which is related to fixed points, is the natural metric for measuring matchings. In order to fit the parameters of a GMM under the Hamming distance, the `dist.name` option has to be used.

```
R> data("perm.sample.med")
R> lgmm(perm.sample.med, dist.name = "hamming")

$mode
[1] 1 2 3 4 5 6
```

```
$theta
[1] 1.24689668 0.29802302 0.01643307 0.79851642 0.55702536 0.79851642
```

Note that the central permutation is given by the identity. Taking into consideration the dispersion parameters, which are a measure of spread, we can analyze how strong the consensus is. For example, the reader can note that θ_1 is the largest spread parameter. This means that the consensus at the first position is large in comparison with the rest and, thus, a large number of permutations in the sample will fix position 1. On the contrary, θ_3 is the smallest, which means that the consensus in this position is the weakest. For a more detailed description of the parameter interpretation see Section 2.2.

The estimations performed so far use approximate algorithms. In order to use an exact estimation algorithm, we can use the `estimation` argument, which can be `approx` (by default) for the approximate learning and `exact` for the exhaustive one.

```
R> my.mm <- lmm(data = perm.sample.med, dist.name = "cayley",
+    estimation = "exact")
R> my.gmm <- lgmm(data = perm.sample.med, dist.name = "cayley",
+    estimation = "approx")
R> my.mm

$mode
[1] 1 2 3 4 5 6

$theta
[1] 0.8364089

R> my.gmm

$mode
[1] 1 2 3 4 5 6

$theta
[1] 1.4087672 0.2876821 0.6931472 1.0986123 0.6190392
```

In some situations, one can have an intuition for the value of the mode of the distribution. It is possible to start the search by an initial guess with the argument `sigma_0_ini`. This can speed the computation up.

```
R> lmm(data = perm.sample.med, sigma_0_ini = c(6, 5, 4, 3, 2, 1),
+    dist.name = "cayley", estimation = "exact")

$mode
[1] 1 2 3 4 5 6

$theta
[1] 0.8364089
```

In the case when the mode is known, the dispersion parameters are obtained with the following functions.

```
R> lmm.theta(data = perm.sample.med, sigma_0 = identity.permutation(6),
+    dist.name = "ulam")
```

```
[1] 0.5890143
```

```
R> lgmm.theta(data = perm.sample.med, sigma_0 = c(2, 1, 6, 5, 3, 4))
```

```
[1] -0.1035352  0.5304489 -0.3297553 -0.2269269 -0.2006699
```

Model sampling Although there are two separate functions for sampling MM and GMM (`rmm` and `rgmm`), they are used in a similar way. The required arguments for the MM (GMM) are the number of permutations to be generated, `n` and the parameters of the distribution, that is σ_0 and θ ($\boldsymbol{\theta}$), and a distance name, which by default is Kendall's-τ.

```
R> rmm(n = 3, sigma0 = my.mm$mode, theta = my.mm$theta)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    5    1    3    2    4    6
[2,]    1    3    2    4    6    5
[3,]    1    4    3    2    5    6
```

```
R> rgmm(n = 3, sigma0 = my.gmm$mode, theta = my.gmm$theta, dist.name = "c")
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    3    1    2    4    5    6
[2,]    1    4    2    5    6    3
[3,]    1    4    6    3    5    2
```

It is also possible to choose among different sampling algorithms using the `sampling.method` argument. The valid options are `distances`, `multistage` or `gibbs`, Section 2.4 for details.

```
R> rmm(n = 4, sigma0 = my.gmm$mode, theta = my.mm$theta, dist.name = "u",
+    sampling.method = "distances")
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    4    1    6    2    5    3
[2,]    4    6    2    3    1    5
[3,]    4    3    2    6    5    1
[4,]    2    5    3    4    6    1
```

Probability The probability of a permutation for a given MM (GMM) is calculated with the `dmm` (`dgmm`) function. The arguments of `dmm` (`dgmm`) are the permutation, the consensus permutation σ_0, dispersion parameter θ ($\boldsymbol{\theta}$) and the name of the distance used, which by default is Kendall's-τ.

```
R> dmm(perm = my.mm$mode, sigma0 = my.mm$mode, theta = my.mm$theta,
+    dist.name = "ulam")

[1] 0.01228094

R> dmm(perm = sigma, sigma0 = my.mm$mode, theta = my.mm$theta,
+    dist.name = "hamming")

[1] 0.00200106

R> dgmm(perm = my.gmm$mode, sigma0 = my.gmm$mode, theta = my.gmm$theta,
+    dist.name = "kendall")

[1] 0.05945101

R> dgmm(perm = sigma, sigma0 = my.gmm$mode, theta = my.gmm$theta,
+    dist.name = "kendall")

[1] 0.0003483458
```

In the following lines we summarize the previous functions using the APA dataset. The MLE for the model parameters are first estimated with the `lmm` (resp. `lgmm`) function for the MM (resp. GMM) under the Kendall's-τ distance and are denoted as `apa.mm` (resp. `apa.gmm`). Then, the log-likelihood of the sample is computed.

```
R> apa.mm <- lmm(data.apa)
R> log.prob <- apply(data.apa, MARGIN = 1, FUN = function(x) {
+    log(dmm(x, apa.mm$mode, apa.mm$theta))
+  })
R> sum(log.prob)

[1] -27408.49

R> apa.gmm <- lgmm(data.apa)
R> log.prob <- apply(data.apa, MARGIN = 1, FUN = function(x) {
+    log(dgmm(x, apa.gmm$mode, apa.gmm$theta))
+  })
R> sum(log.prob)

[1] -27405.41
```

Note that the log-likelihood of the MLE for the GMM, with n parameters, fits the APA dataset slightly better than its particularization with two parameters, the MM.

Regarding the inference operators, **PerMallows** includes functions to compute the expectation of the distance (resp. its decomposition vector) under the MM (resp. GMM). The function `expectation.mm` computes the expectation of the distance under an MM of a given dispersion parameter and number of items, while `expectation.mm` computes the expectation of the distance decomposition vector given a multidimensional dispersion parameter.

```
R> expectation.mm(theta = 3, perm.length = 9, dist.name = "ulam")

[1] 1.812466

R> expectation.gmm(c(1.1, 2, 1, 0.2, 0.4), dist.name = "cayley")

[1] 0.6246747 0.3512144 0.5246331 0.6208475 0.4013123
```

Finally, **PerMallows** includes the computation of marginal distribution for both MM and GMM under the Hamming distance. The set of fixed and unfixed points is represented in the distance decomposition vector, so $H_j(\sigma) = 0$ means that j is a fixed point, $H_j(\sigma) = 1$ that j is an unfixed point and `NA` means that j is unknown. The following command computes the marginal distribution of those permutations having fixed points at positions 2 and 4 and unfixed points at position 1.

```
R> marginal(h = c(1, 0, NA, 0, NA), theta = c(1.1, 2, 1, 0.2, 0.4))

[1] 0.0808545
```

Handling large permutations under the Ulam distance The most computationally expensive version of any function is usually that concerning the Ulam distance. **PerMallows** can handle MM under the Ulam distance with permutations of 80 items perfectly in any regular personal computer. For larger permutation sizes, **PerMallows** offers the possibility of preprocessing a problem, so the operations –such as counting and generating permutations, learning, sampling– are performed very fast. This preprocess consists of generating files relative to each of the partitions of `perm.length`.

In the case where one expects to work repeatedly with a model under the Ulam distance of a particular permutation length larger than 80 (independently of the value of θ), it is a good idea to run the preprocess. The preprocess is called with the following function:

```
R> generate.aux.files(perm.length = 6)

[[1]]
[1] 6
```

In order to use the preprocessed information, it is only necessary to use the `disk=TRUE` argument to the above defined functions. Some examples are as follows.

```
R> count.perms(perm.length = 6, dist.value = 4, dist.name = "ulam",
+    disk = TRUE)

[1] 131

R> lmm(data = perm.sample.med, dist.name = "ulam", disk = TRUE)

$mode
[1] 2 5 1 3 4 6

$theta
[1] 0.8899959

R> rmm(n = 3, sigma0 = identity.permutation(6), theta = 1, dist.name = "u",
+    disk = TRUE)

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    4    5    1    2    3    6
[2,]    1    4    3    5    2    6
[3,]    1    5    6    2    4    3
```

4. Conclusions

This paper describes an R package for dealing with probability distributions over permutations spaces, **PerMallows**. The models used are the Mallows (MM) and generalized Mallows (GMM) models. Both models require a distance for permutations; **PerMallows** considers the Kendall's τ, Cayley, Hamming and Ulam metrics.

The **PerMallows** package is aimed at being a compilation of resources for working on MM and GMM. It provides functions for the exact and approximate estimation of the parameters of a collections of permutations, for simulating from a given distribution or calculating the density function.

Efficient algorithms for reasoning on permutation spaces can not be given without taking into consideration the particular nature of permutations. Therefore, the core of the **PerMallows** package consists of several functions and operators for permutations such as the factorization of permutations, the random generation of permutations, counting the number of permutations at a given distance, etc.

We expect the **PerMallows** package to be helpful to every kind of user, from the novice in the field of permutations and/or probability models for permutation spaces to the advanced users.

Acknowledgments

This work has been partially supported by the Saiotek and IT-609-13 programs (Basque Government) and TIN2013-41272P (Spanish Ministry of Science and Innovation).

References

Ali A, Meila M (2012). "Experiments with Kemeny Ranking: What Works When?" *Mathematical Social Sciences*, **64**(1), 28–40. doi:10.1016/j.mathsocsci.2011.08.008.

Arora R, Meila M (2013). "Consensus Ranking with Signed Permutations." In *Artificial Intelligence and Statistics (AISTATS)*, volume 31 of *JMLR Proceedings*, pp. 117–125.

Babington Smith B (1950). "Discussion on Professor Ross's Paper." *Journal of the Royal Statistical Society*, **12**, 153–162.

Bader M (2011). "The Transposition Median Problem Is NP-Complete." *Theoretical Computer Science*, **412**(12-14), 1099–1110. doi:10.1016/j.tcs.2010.12.009.

Bóna M (2004). *Combinatorics of Permutations.* Chapman & Hall/CRC Press, Boca Raton.

Borda J (1781). *Memoire sur les Elections au Scrutin.* Histoire de l'Academie Royal des Sciences.

Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G (2005). "Learning to Rank Using Gradient Descent." In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 89–96. ACM.

Caragiannis I, Procaccia AD, Shah N (2013). "When Do Noisy Votes Reveal the Truth?" In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce*, EC '13, pp. 143–160. ACM, New York. doi:10.1145/2482540.2482570.

Ceberio J, Mendiburu A, Lozano JA (2011). "Introducing the Mallows Model on Estimation of Distribution Algorithms." In *International Conference on Neural Information Processing (ICONIP)*, 23-25. Shanghai.

Cheng W, Hüllermeier E (2009a). "A New Instance-Based Label Ranking Approach Using the Mallows Model." In *Advances in Neural Networks (ISNN)*, volume 5551 of *Lecture Notes in Computer Science*, pp. 707–716. Springer-Verlag. URL http://dblp.uni-trier.de/db/conf/isnn/isnn2009-1.html{#}ChengH09.

Cheng W, Hüllermeier E (2009b). "A Simple Instance-Based Approach to Multilabel Classification Using the Mallows Model." In *Workshop Proceedings of Learning from Multi-Label Data*, pp. 28–38. Bled, Slovenia.

Cohen WW, Schapire RE, Singer Y (1998). "Learning to Order Things." In *Advances in Neural Information Processing Systems (NIPS)*, NIPS '97, pp. 451–457. MIT Press, Cambridge. ISBN 0-262-10076-2.

Coppersmith D, Fleischer LK, Rurda A (2010). "Ordering by Weighted Number of Wins Gives a Good Ranking for Weighted Tournaments." *ACM Transactions on Algorithms*, **6**(3), 1–13. ISSN 1549-6325. doi:10.1145/1798596.1798608.

Critchlow DE, Fligner MA, Verducci JS (1991). "Probability Models on Rankings." *Journal of Mathematical Psychology*, **35**, 294–318. doi:10.1016/0022-2496(91)90050-4.

D'Elia A, Piccolo D (2005). "A Mixture Model for Preferences Data Analysis." *Computational Statistics & Data Analysis*, **49**(3), 917–934. ISSN 0167-9473. doi:10.1016/j.csda.2004.06.012.

Diaconis P (1988). *Group Representations in Probability and Statistics*. Institute of Mathematical Statistics.

Diaconis P (1989). "A Generalization of Spectral Analysis with Application to Ranked Data." *The Annals of Statistics*, **17**(3), 949–979. doi:10.1214/aos/1176347251.

Dwork C, Kumar R, Naor M, Sivakumar D (2001). "Rank Aggregation Methods for the Web." In *International Conference on World Wide Web*, WWW '01, pp. 613–622. ACM, New York. doi:10.1145/371920.372165.

Farah M, Vanderpooten D (2007). "An Outranking Approach for Rank Aggregation in Information Retrieval." In *Conference on Research and Development in Information Retrieval (ACM SIGIR)*, SIGIR '07, pp. 591–598. ACM, New York. doi:10.1145/1277741.1277843.

Féray V (2013). "Asymptotics of Some Statistics in Ewens Random Permutations." *Electronic Journal of Probability*, **18**(76), 1–32. doi:10.1214/ejp.v18-2496.

Fligner MA, Verducci JS (1986). "Distance Based Ranking Models." *Journal of the Royal Statistical Society B*, **48**(3), 359–369.

Fligner MA, Verducci JS (1988). "Multistage Ranking Models." *Journal of the American Statistical Association*, **83**(403), 892–901. ISSN 01621459. doi:10.2307/2289322.

Furnkranz J, Hullermeier E (2013). "Preference Learning and Ranking." *Machine Learning*, **93**(2–3). doi:10.1007/s10994-013-5414-z.

Gnedin A, Olshanski G (2012). "The Two-Sided Infinite Extension of the Mallows Model for Random Permutations." *Advances in Applied Mathematics*, **48**(5), 615–639. ISSN 0196-8858. doi:10.1016/j.aam.2012.01.001.

Gregory E (2012). ***RMallow**: Fit Multi-Modal Mallows' Models to Ranking Data*. R package version 1.0, URL https://CRAN.R-project.org/package=RMallow.

Hardy GH, Ramanujan SA (1918). "Asymptotic Formulae in Combinatory Analysis." *Journal London Mathematical Society*, **17**, 75–115. doi:10.1112/plms/s2-17.1.75.

Hatzinger R, Dittrich R (2012). "**prefmod**: An R Package for Modeling Preferences Based on Paired Comparisons, Rankings, or Ratings." *Journal of Statistical Software*, **48**(10), 1–31. doi:10.18637/jss.v048.i10.

Irurozki E (2014). *Sampling and Learning Distance-Based Probability Models for Permutation Spaces*. Ph.D. thesis, University of the Basque Country.

Lee PH, Yu PLH (2012). "Mixtures of Weighted Distance-Based Models for Ranking Data with Applications in Political Studies." *Computational Statistics & Data Analysis*, **56**(8), 2486–2500. ISSN 0167-9473. doi:10.1016/j.csda.2012.02.002.

Lee PH, Yu PLH (2015). ***pmr**: Probability Models for Ranking Data*. R package version 1.2.5, URL https://CRAN.R-project.org/package=pmr.

Lu T, Boutilier C (2011a). "Learning Mallows Models with Pairwise Preferences." In *International Conference on Machine Learning (ICML)*, pp. 145–152.

Lu T, Boutilier C (2011b). "Vote Elicitation with Probabilistic Preference Models: Empirical Estimation and Cost Tradeoffs." In RI Brafman, FS Roberts, A Tsoukiàs (eds.), *Algorithmic Decision Theory*, volume 6992 of *Lecture Notes in Computer Science*, pp. 135–149. Springer-Verlag. doi:10.1007/978-3-642-24873-3_11.

Luce RD (1959). *Individual Choice Behavior*. John Wiley & Sons, New York.

Mallows CL (1957). "Non-Null Ranking Models." *Biometrika*, **44**(1–2), 114–130. doi:10.1093/biomet/44.1-2.114.

Mandhani B, Meila M (2009). "Tractable Search for Learning Exponential Models of Rankings." *Journal of Machine Learning Research*, **5**, 392–399.

Mao Y, Lebanon G (2008). "Non-Parametric Modeling of Partially Ranked Data." *Journal of Machine Learning Research*, **9**, 2401–2429.

Meila M, Chen H (2010). "Dirichlet Process Mixtures of Generalized Mallows Models." In *Uncertainty in Artificial Intelligence (UAI)*, pp. 285–294.

Meila M, Le Bao (2008). "Estimation and Clustering with Infinite Rankings." In *Uncertainty in Artificial Intelligence (UAI)*, pp. 393–402. AUAI Press, Corvallis, Oregon.

Mueller C, Starr S (2013). "The Length of the Longest Increasing Subsequence of a Random Mallows Permutation." *Journal of Theoretical Probability*, **26**(2), 514–540. ISSN 0894-9840. doi:10.1007/s10959-011-0364-5.

Murphy TB, Martin D (2003). "Mixtures of Distance-Based Models for Ranking Data." *Computational Statistics & Data Analysis*, **41**(3-4), 645–655. ISSN 0167-9473. doi:10.1016/s0167-9473(02)00165-2.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Schader M (1991). *Analyzing and Modeling Data and Knowledge*. Chapman & Hall. doi:10.1007/978-3-642-46757-8.

Sloane NJA (2009). "On-Line Encyclopedia of Integer Sequences." URL http://oeis.org/.

Strobl C, Wickelmaier F, Zeileis A (2011). "Accounting for Individual Differences in Bradley-Terry Models by Means of Recursive Partitioning." *Journal of Educational and Behavioral Statistics*, **36**(2), 135–153. doi:10.3102/1076998609359791.

Sun M, Lebanon G, Kidwell P (2012). "Estimating Probabilities in Recommendation Systems." *Journal of the Royal Statistical Society C*, **61**(3), 471–492. doi:10.1111/j.1467-9876.2011.01027.x.

Thurstone LL (1927). "A Law of Comparative Judgment." *Psychological Review*, **34**(4), 273–286. ISSN 0033-295X. doi:10.1037/h0070288.

Turner H, Firth D (2012). "Bradley-Terry Models in R: The **BradleyTerry2** Package." *Journal of Statistical Software*, **48**(9), 1–21. doi:10.18637/jss.v048.i09.

Ziegler A, Christiansen E, Kriegman D, Belongie S (2012). "Locally Uniform Comparison Image Descriptor." In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1–9. URL http://books.nips.cc/papers/files/nips25/NIPS2012_0012.pdf.

Affiliation:

Ekhine Irurozki, Borja Calvo, Jose A. Lozano
Intelligent Systems Group
Department of Computer Science and Artificial Intelligence
Faculty of Computer Science
University of the Basque Country
20008 Donostia, Spain
E-mail: ekhine.irurozqui@ehu.es, borja.calvo@ehu.es, ja.lozano@ehu.es
URL: http://www.sc.ehu.es/ccwbayes/isg/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
August 2016, Volume 71, Issue 12 *Submitted:* 2014-07-11
doi:10.18637/jss.v071.i12 *Accepted:* 2015-08-07